This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

Claim 1 (currently amended): A method for processing software instructions comprising:

decomposing a macroinstruction into a plurality of microinstructions~~,~~;

forcing the parallel issue ~~issuing all~~ of at least two of the plurality of microinstructions simultaneously~~, in parallel,~~;

executing ~~all of~~ the at least two ~~plurality of~~ microinstructions simultaneously, in lockstep using functional units in a floating point unit~~,~~;

determining whether an exception occurs in any of the microinstructions, before writing results of the executing to result registers~~,~~;

if an exception occurs in any of the microinstructions, canceling all of the microinstructions and preventing the results of the executing from being written to the result registers~~,~~; and

if no exception occurs in any of the microinstructions, writing the results of the executing to the result registers.

Claim 2 (previously canceled)

Claim 3 (previously amended): The method of claim 1, wherein the microinstructions are executed on separate execution units, but appear as though they were executed on a single execution unit.

Claim 4 (previously amended): The method of claim 1, wherein all of the microinstructions are executed on the same clock cycle.

Claim 5 (previously amended): The method of claim 1, wherein the microinstructions are executed over multiple clock cycles.

Claim 6 (canceled)

Claim 7 (currently amended): The method of claim ~~6~~ 1, wherein the system allows a single instruction to operate on multiple single-precision ~~("SP")~~ floating-point ~~("FP")~~ values.

Claim 8 (previously amended): The method of claim 1, further comprising updating a flag based upon a result of the execution of the microinstructions.

Claim 9 (previously amended): The method of claim 1, further comprising,

if an unmasked exception occurs, canceling the execution of all of the plurality of microinstructions, without regard to the relative ages of

each of the plurality of microinstructions, and invoking a microcode handler, and

if an unmasked exception does not occur, updating at least one exception flag by independently generating a logical OR of exceptions for a plurality of functional units.

Claim 10 (currently amended): A method for processing software instructions comprising:

providing two microinstructions to emulate a high-half and a low-half SSE operation~~s~~;

forcing the high-half and low-half operations to issue in parallel~~,~~;

dispatching the high-half and low-half operations simultaneously to a first ~~FP~~ floating point unit and to a second ~~FP~~ floating point unit, respectively~~,~~;

executing the high-half and low-half operations simultaneously, in lockstep,

generating a signal from an emulator's hardware~~,~~;

sending the signal to the first and second ~~FP~~ floating point functional units,

determining whether an exception is taken in either the first or the second ~~FP~~ floating point unit~~s~~;

if an exception is taken in either the first or second ~~FP~~ floating point unit, preventing results from the high-half and low-half operations from being written to result registers~~,~~;

canceling both the high-half and low-half operations~~,~~; and

updating MXCSR flags based upon the results of the first and second ~~FP~~ floating point units.

Claim 11 (currently amended): The method of claim 10, wherein the flushing of a result in the other ~~FP~~ floating point unit does not depend upon the relative ages of the two microinstructions.

Claim 12 (currently amended): A computer system comprising:

a processor comprising~~,~~;

a floating point unit comprising a plurality of functional units adapted to execute microinstructions;

a ROM;

a plurality of floating point registers;

wherein the processor is configured to emulate an instruction set by:

decomposing a macroinstruction into a plurality of microinstructions;

forcing the parallel issue ~~issuing~~ of at least two ~~all~~ of the plurality of microinstructions simultaneously~~, in parallel,~~ to the functional units~~,~~;

determining whether an exception occurs in any of the functional units~~,~~;

setting result registers for results of each of the functional units only if no exception occurs in any of the functional units~~,~~; and

if an exception occurs in any of the microinstructions, canceling all of the microinstructions and preventing the setting of result registers for all of the functional units.

Claim 13 (previously amended): The computer system of claim 12, wherein the processor is further configured to emulate the instruction set by executing all of the microinstructions.

Claim 14 (previously amended): The computer system of claim 13, wherein the microinstructions are executed on separate execution units, but appear as though they were executed on a single execution unit.

Claim 15 (previously amended): The computer system of claim 14, wherein the processor is further configured to emulate an instruction set by updating a flag based upon a result of the execution of the microinstructions.

Claim 16 (previously canceled)

Claim 17 (canceled)

Claim 18 (currently amended): The computer system of claim 17, further comprising an ~~FP~~ floating point register having 82 bits, wherein the computer system uses two ~~FP~~ floating point registers to emulate four 32-bit single-precision, floating point values in an SSE register.

Claim 19: The method of claim 1, wherein the step of issuing comprises forcing the microinstructions to issue simultaneously, in lockstep with each other, and wherein the step of canceling comprises canceling all of the plurality of microinstructions without regard to the relative ages of the microinstructions and without using a backoff mechanism.

Claim 20 (currently amended): The method of claim 10, wherein the step of forcing the high-half and low-half operations to issue in parallel comprises causing the high-half and low-half operations to execute simultaneously in lockstep with each other, and wherein the step of flushing a result comprises canceling each of the high-half and low-half operations if an exception is taken in either the first or second ~~FP~~ floating point unit.

Claim 21 (previously added): The method of claim 1,

wherein the step of executing comprises executing using a plurality of functional units of a floating point unit,

further comprising:

generating a signal using emulation hardware, wherein the signal indicates that the functional units are emulating an SSE instruction; and

sending the signal to the functional units, and

wherein the step of determining comprises determining after the signal is sent.

Claim 22 (previously added): The system of claim 12, wherein the processor is further configured to emulate an instruction set by:

generating a signal using emulation hardware, wherein the signal indicates that the functional units are emulating an SSE instruction; and

sending the signal to the functional units, and

wherein the step of determining comprises determining after the signal is sent.